International Journal of Electrical and Electronics Engineering (IJEEE)
ISSN(P): 2250-155X; ISSN(E): 2278-943X
Vol. 6, Issue 2, Apr 2016, 27-36

TRANS STELLAR
•Journal Publications •Research Consultancy

NET TRAFFIC BALANCER USING SINGLE BOARD COMPUTER

DHRUVA R. RINKU[1] & M. ASHA RANI[2]

[1] *Associate Professor, Department of Electronics & Communication CVR College of Engineering, Hyderabad, India*

[2]*Head of the Department of Electronics & Communication JNT University, Hyderabad, India*

ABSTRACT

Internet technology has become the backbone of information management and usage of it has been growing multifold, demanding a solution to cater the service more effectively as well as efficiently. One major requirement is how well the servers would be able to cater the requests from multiple clients during peak demands. Many solutions exist to meet this requirement, but popular solution is clustering servers and sharing the load among them. This paper explores one such opportunity, using the most cost effective solution with minimal hardware and software resources.

KEYWORDS: Client, Server Cluster, Load Share

Received: Feb 15, 2016; **Accepted:** Feb 26, 2016; **Published:** Mar 02, 2016; **Paper Id.:** IJEEERAPR201603

Original Article

INTRODUCTION

The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may even reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Web server is software that provides data services that functions accept HTTP requests from clients, known as a web browser and sends the results back in the form of web pages that usually form an HTML document.

Apache web server is the most widely used one in the Internet. Technical support resources for Apache are available on multiple websites around the world. Apache Web Server has the advantage of some of the considerations; Apache is included in the freeware category. Apache is able to operate on various operating system platforms and easy to set the configuration.

Traffic to a web site can vary dramatically. At the same time it is highly desirable that a web site is reactive. To provide crisp interaction on thin clients, 150 milliseconds has been suggested as an upper bound on response time.

Unfortunately, the popular Apache Web server is limited in its capabilities to be reactive under varying traffic. To work around this problem, we design in this paper web cluster with the Apache Web server.

Clustering with server is popular [1], in which group of independent servers (usually in close proximity to one another) interconnected through a dedicated network to work as one centralized data processing resource. Clusters can be used as web servers [2]. Many web sites, in turn the servers which serve the sites get too much

traffic to be run on a single server, so several servers have to be used. Requests from web browsers (clients) are received by a node called a load balancer, which forwards requests to worker servers. The load balancer then forwards responses from servers back to the clients.

One way to ensure quick and predictable user response is through load balancing. A single server's ability to service clients is bound by multiple factors, including CPU utilization, memory capacity, bandwidth capacity, and I/O rate. Load balancing at the cluster level allows connections to be distributed to servers with the least load. Typical implementations of load balancing deal with round robin balancing.

Server load balance is achieved by continuous monitoring of the load of each co server and dynamically redirecting ongoing or new service requests to available servers in such a manner that the end user experiences the lowest delay and distortion when one or more servers are overloaded. The controller application manages two basic functions, namely server load monitoring and server selection/ flow updating. Here, Server load monitoring continuously checks whether the server load is within a predefined capacity threshold. Server selection and flow updating function is invoked when an overload condition is detected.

Figure 1: Load Balancer with Backend Servers

Proposal

In this paper server load balancer system is implemented.

RaspberryPis are small and inexpensive Single Board Computers, One popular use of RaspberryPi SBCs is building clusters[4]. So it is easier to use them to build a cluster using Raspberry Pis than it would be with PCs. Although a Pi cluster may not be that powerful as compared to a full fledged computer, it is effective when it comes to net traffic load balancing.

The proposed system comprises a cluster of two RaspberryPi SBCs. Here, load balancer is also implemented using a third Raspberry Pi, which is responsible for balancing the request posted by client to the co-servers. If the requests are more on one server than threshold, then request will be transferred to the second co server to maintain the quality of service and to reduce the load on the first server. Load balancer is capable of performing multiple complex instructions by distributing workload across all connected servers.

To distinguish between the two co-servers, one server is interfaced with a camera and another with a temperature monitoring and display. Though, in the real life situation, both the co-servers would serve the same information. This implementation also provides an opportunity to explore various capabilities of RaspberryPi boards with different

applications.

Implementation Methodology

Each Pi SBC is configured by porting Raspbian Operating system, OS image file is taken from raspberrypi.org site and ported on 4GB SD card with the use of NOOBS [10]. One Pi SBC has been used to implement video capture image co server. This co server provides images which is captured by CMOS camera that is connected to the board. QTcreator IDE[17],[18] is used for Graphical user Interface to create widgets for web pages of servers. OpenCV[14] open source has been used for image related processing operation. Another Pi SBC has been used to implement second server which monitors and displays temperature using a sensor child card. Here I2C protocol has been used to communicate with Pi SBC, because Pi does not have on chip ADC, PCF8591 A/D converter board has been used to implement this. Third Pi SBC is used to implement load balancer, which is implemented with scripting language. On three SBCs Apache 2 web server is loaded [14].

All the Pies in the cluster are configured with static IP addresses. The IP addresses used in the cluster are on the 192.168.1.xxx subnet.

Raspberry Pi 2 SBC:

Figure 2: Raspberry Pi Board

Table 1: Specifications of Board

Ethernet	100 Mbps
USB	4 x USB 2.0
Video out	HDMI 1.4
Audio	2 x analog
CPU	900MHz quad-core ARM Cortex-A7
card slot	Micro SD

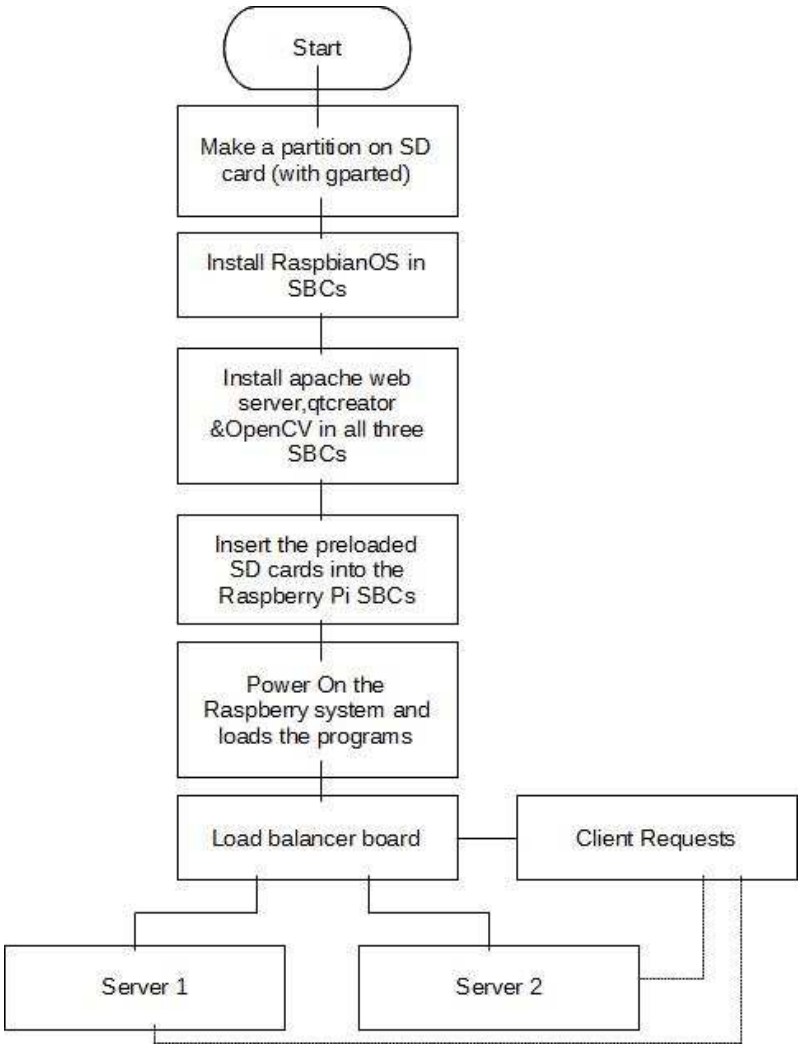

Figure 3

This implementation uses three RPi2s, one Ethernet router. A power supply with three micro USB cables are provided to power up the boards. Three SD cards are needed with Pi boards as secondary memory containing the Operating System, and other needed software. The boards are interconnected to regular monitors using HDMI-VGA cables, and USB keyboards and mice.

First to make partition in SD card Gparted application software has been used, SD card gets two partitions which are Bootloader and FAT32 file system.

An image comprising Raspbian OS, qtCreator, and OpenCV has been made and ported into all three SCBs, with the use of following system command:

sudo bash

% dd if = pi2.img of = /dev/sdb bs=4M

On each card Apache web server got installed by command:

sudo apt-get install apache2

Each Pi SBC has been assigned with with a static IP address by changing IP address in commandline.txt in boot

folder. The server 1 with IP address is 192.168.1.91 displays live video image captured by CMOS camera.: video streaming code has been loaded, which is implemented with Qt creator IDE and OpenCV function :

Figure 4: Server 1 - Video Streaming

```
{
if(!cap->isOpened())
return ;
Mat frame,frame1;
cap->read(frame1);
cvtColor(frame1 ,frame ,CV_BGR2RGB);
QImage image((uchar *)frame.data,
frame.cols, frame.rows, frame.step, QImage::Format_RGB888);
ui->image->setPixmap(QPixmap::fromImage(image));
imwrite("/var/www/img.jpg",frame);
qDebug("save");
}
```

The server 2 with IP address is 192.168.1.92 displays the temperature sensor output.

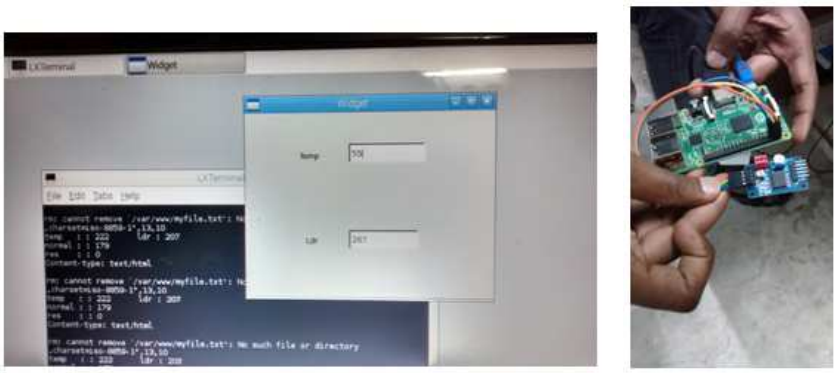

Figure 5: Server 2 - Temperature Monitor

Snippet of source code is*:*

```
int Widget::readInput(int fd, int reg)
{
wiringPiI2CReadReg8(fd, reg);
return wiringPiI2CReadReg8(fd, reg);
}
void Widget::sensor()
{
// fd=open("data.txt", O_RDWR | O_NONBLOCK);
//camera();
wiringPiSetupGpio();
int dacModule = wiringPiI2CSetup(0x48);
if (dacModule < 0)
{
cout << "I2C Setup Error" << endl;
//return 0;
}
```

on SBCcard with IP address 192.168.1.90 : Load Balancer is implemented. part of code is:

```
</Directory>
ProxyRequests Off
<Proxy> balancer://rpicluster
BalancerMember http://192.168.1.91:80
BalancerMember http://192.168.1.92:80
AllowOverride None
Order allow,deny
allow from all
ProxySet lbmethod=byrequests
</Proxy>
<Location /balancer-manager>
SetHandler balancer-manager
```

Order allow,deny

allow from 192.168.1

</Location>

ProxyPass /balancer-manager !

ProxyPass / balancer://rpicluster/

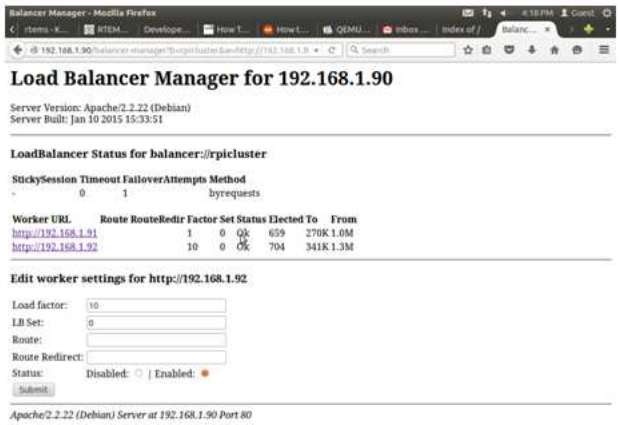

Figure 6: Load Balancer

RESULTS

Figure 7

Clients are given the IP address of Load balancer which is 192.168.1.90. The load balancer re-route the requests to appropriate server based on the load of it. To see the webpage of any server in the cluster one has to type balancer IP address (in this case, 192.168.1.90). Load balancer node re-routes the requests to either server, hence with the same IP address client 1 is getting the data from server 1 (video stream) and client 2 is getting the data from server 2 (temperature monitor) as shown in figures 6 and 7

CONCLUSIONS

Clustered internet hosting provides optimal service to the client requests with replicated web servers. Multiple clients may request the information from the web server through IP address 192.168.1.90. The load balancer receives the

request and checks the load on each server and roots the request to the server with lesser load. The IP addresses of the replicated servers are transparent to the user. Hence, with this approach multiple servers can be setup with replicated data and can serve more clients efficiently than one server can handle.

FUTURE SCOPE

The authors tried to touch upon the possibilities of implementing a net traffic balancer with minimal resources. The server clusters can be multiplied to get better results and full fledged computers can replace the SBCs to achieve commercial and large scale implementations.

ACKNOWLEDGEMENTS

The author is grateful to her Ph.D. Guide Mrs. Dr. M. AshaRani, HOD Professor (Jawaharlal Nehru Technology University) for her encouragement and guidance to present/publish this paper. I acknowledge the diligent efforts of Mr. YVS Ravikanth in assisting towards implementation of this idea.

REFERENCES

1. http://docs.gigaspaces.com/xap102/apache-load-balancer-agent.html
2. https://www.youtube.com/load balancer-manager
3. http://www.networkworld.com/article/2225683/cisco-subnet/raspberry-pi-as-a-network-monitoring-node.html
4. http://raspberrywebserver.com/raspberrypicluster/raspberry-pi-cluster.html
5. http://networkgeekstuff.com/networking/load-balancing-traffic-on-mikrotik-router-tutorial
6. About Raspberry pi 2 ,http://www.raspberrypi.org
7. Raspberry Pi hardware information,http://elinux.org/RPI_Hardware
8. How SoC works,http://www.androiddauthority.com/how-it-works-systems-on-a-chip-soc-93587
9. Bootprocess,http://thekandyancodewordpress.com/2013/09/21/how-the-raspberry-pi-boots-up/
10. NOOBS , http://www.raspberrypi.org/introducing-noobs/
11. List of available OS, http://elinux.org/RPI_Distributions
12. http://www.linux-projects.org/modules/sections/index.php?op=viewarticle&artid=14
13. http/;//wikipedia.org/wiki/Apache web server
14. docs.opencv.org/doc/tutorials/tutorials.html
15. www.learnopencv.com
16. how-to-link-opencv-in-qtcreator
17. doc.qt.io/qtcreator/creator-tutorials.html
18. https://wiki.qt.io/Basic_Qt_Programming_Tutorial
19. The Efficient load balancing in the parallel Computer International Journal of Advanced Research in Computer and Communication Engineering.Vol. 2, Issue 4, April 2013
20. Small Data Center using Raspberry Pi 2 for Video Streaming,by P.J.E. Velthuis.

21. *Cassandra. Welcome to apache cassandra. http://cassandra.apache.org/, 2015*
22. *M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali. Cloud computing: Distributed internet computing for it and scientific research. Internet Computing, IEEE, 13(5):10–13, 2009. doi: 10.1109/MIC.2009.103*
23. *Dan and D. Sitaram. Load balancing in video-on-demand servers by allocating buffer to streams with successively larger buffer requirements until the buffer requirements of a stream can not be satisfied, Aug. 6 1996. US Patent 5,544,327.*
24. *M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis,and A. Vakali. Cloud computing: Distributed internet computing for it and scientific research.Internet Computing, IEEE, 13(5):10–13, 2009. doi: 10.1109/MIC.2009.103.*
25. *P. Abrahamsson, S. Helmer, N. Phaphoom,L. Nicolodi, N. Preda, L. Miori, M. Angriman,J. Rikkila, X. Wang, K. Hamily, et al. Affordable and energy-efficient cloud computing clusters: Thebolzano raspberry pi cloud cluster experiment. In Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, volume 2, pages 170–175. IEEE, 2013. doi: 10.1109/CloudCom.2013.121. URL 10.1109/CloudCom.2013.121.*
26. *R.-Y. Shieh, G. Tsai, Y.-A. Chen, and C.-L. Chang. Externally connection type usb2. 0 interface flash cardreader.https://www.google.com/patents/US20020185533,2001. US Patent App. 09/874,303*
27. *QHM49LM web camera data sheet*
28. *PCF8591 8 bit A/D and D/A converter datasheet*